# Assignment 2

In this assignment, you will write code in CarSim (in your controller) to develop a path to the target and follow it with your vehicle. To do this, you should first discretize the space (in your own code) by imagining a grid over it, and then compute a short path on the grid. More precisely:

- You will write a car-controller that implements the CarController interface.
- When the simulator calls your `init()` method, you will be passed the initial location, the desired end-location (where the target is), and a list of obstacles.
- Write code in the `init()` method to compute a path, and store that path in a local variable or data structure. Later, you will use this path in the `move()` method.
- When the simulator calls your `move()` method, you will have the opportunity to follow the path you pre-computed earlier in the `init()` method.
- You should test your code with each of the "scenes" and a more complex scene of your making. You can do this by modifying Scene-1 in `CarGUI`. Design your scene so that at least one part of the optimal path navigates through a channel narrow enough to barely allow your vehicle. We will also create an additional (more complex) scene for this assignment.

So, how should you compute a path? A simple approach is to throw an imaginary grid over the space and treat the grid as a graph.

- You can use any shortest-path algorithm, such as Dijkstra's algorithm. However, for a uniform grid, a simpler, more intuitive approach will work just fine.
- You need to make sure the path is wide enough to accomodate your vehicle.
- The grid needs to be fine enough to find a path between obstacles (as in Scene-4).
- Notice that the CarController interface has a `draw()` method. You should use this for debugging. To achieve this, draw your path on the canvas (converting to Java coordinates) and observe how your car follows the path.
- Note: Undergrad students can use the unicycle. Grads should handle the unicycle and the Dubin car (non-accelerative versions), and therefore submit two car controllers.
- The dimensions of the cars are likely to be useful. The unicycle is an oval that fits into a rectangle of length 30 and width 16. The simple car is of length 40 and width 30. The Dubin car dimensions are: length=40, width=20 (wheel spacing), wheel radius=5, wheel thickness=4. All are in Java pixels, the same units used for the scenes and obstacles.
- Ignore Scene-5 (that's for another assignment).

The ultimate goal is to get to the destination and as fast as possible and stop there. Let's consider some of the relevant details:

- Your `move()` method should strive to move your car along the path you compute.
- To do this, you will need to know "where you are" at any given time. For this purpose, your car is given a SensorPack in the `init()` method, which you should store in a variable. You can call the `getX()` and `getY()` methods of the `SensorPack` to find out your current position. Examine the code in `BasicSensorPack` (an implementation of the `SensorPack` interface) and you'll see that the actual (x,y) values are simply passed back through these methods.

Finally, a challenge: for bonus points, see if you can get an accelerative version to reach the target significantly faster than the equivalent non-accelerative version.

**Submission:**

- Download the latest copy of carSim.jar to get the updates made to the simulator since the first assignment.
- Make sure all *your* Java classes are named such have your username is embedded in each class name. This is to avoid name clashes with other students. Thus, if your username is `karel`, you can call your controller `KarelCarController.java`.
- **Important**: henceforth, please include a README in every submission. This file is where you will include relevant documentation, special instructions on running your code (if applicable), and so on. For example, in this assignment, you will need to explain your path-finding algorithm and how you make sure the path is wide enough to accomodate your vehicle.
- Since you are modifying the simulator (in `CarGUI`) to add your own Scene-1, include the modified simulator in your submission.
- Name your zip file `karel2.zip` (for username `karel`).
- Upload your file in Blackboard under Assignment 2.